**Git fundamental (essential only)**

ผู้อ่านกลุ่มเป้าหมาย

บทความนี้มุ่งเป้าหมายไปที่นักพัฒนาSoftwareที่รู้จักVersion Control Systemและต้องการเข้าใจหลักความคิดพื้นฐานของGitเพื่อให้มีความรู้พื้นฐานและสามารถประยุกต์ความรู้นั้นการดึงความสามารถสูงสุดจากGit commandและมีความเข้าใจพื้นฐานสำหรับการเรียนรู้ Git Frameworkอื่นๆต่อไป

เอกสารนี้แตกต่างเอกสารที่มีอยู่ก่อนหน้าอย่างไร

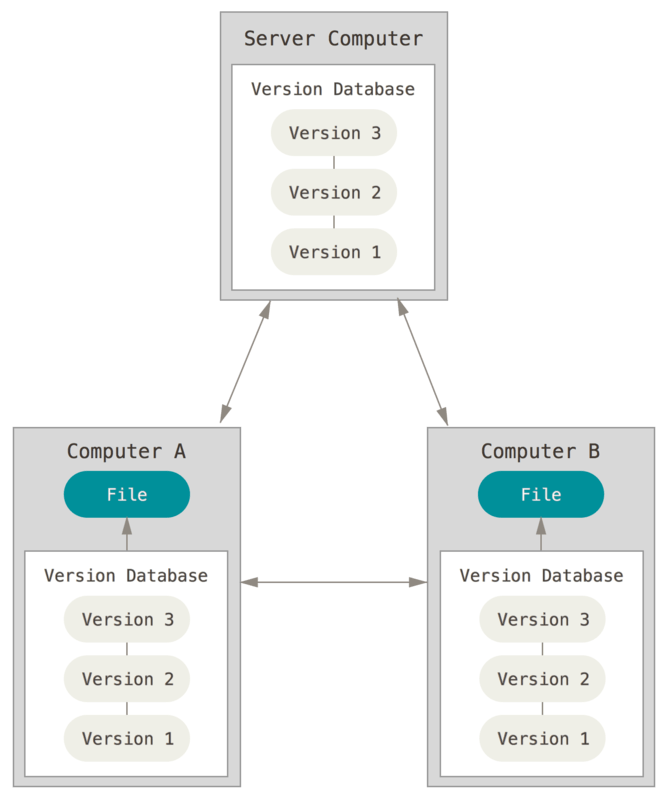
เอกสารอื่นๆที่มีอยู่ก่อนหน้าไม่ได้อธิบายหลักการความคิดขั้นพื้นฐานของGitแต่อธิบายการใช้งาน Command, การใช้งานFrameworkจากvendorต่างๆ และ อธิบายWorkflowของแต่ละFrameworkแทน

ความสำคัญของการเข้าใจพื้นฐาน Git

Git Framework เป็นสิ่งที่องค์กรหลายๆองค์กรใช้ในการควบคุมการพัฒนาProject แต่ละองค์กรใช้Frameworkแตกต่างกัน การมีความรู้พื้นฐานGitจะทำให้การเรียนรู้และการใช้งานFrameworkเหล่านั้นง่ายและรวดเร็วมากขึ้น

Git Idea

- Distribute Version Control



Gitเป็นVersion Controlที่มีการทำงาน

แบบDistribute นั้นหมายความว่าในแต่ละ

Local machineจะมีProject directory

เหมือนกับเครื่องอื่นๆถ้าถูกดึงข้อมูล ณ เวลา

เดียวกัน

* Nearly Every Operation Is Local

เนื่องจากทำงานเกือบทั้งหมดจะอยู่บนLocalการพึ่งพาNetworkเพื่อการทำงานGitจึงน้อย

มาก

* Only add data

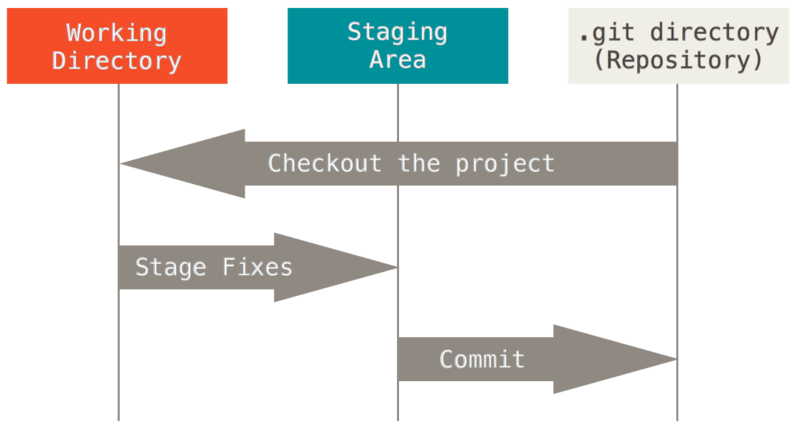
โดยมากGit จะไม่ลบData Git จึงมีความสามารถที่จะดึง ข้อมูลที่เหมือนถูกลบไปแล้ว

กลับมาได้

Gitจะเก็บProjectตัวเต็มเอาไว้จึงไม่จำเป็นที่จะต้องไปโหลดการข้อมูลเปลี่ยนแปลงมาจากแหล่งอื่น เฉพาะไฟล์ที่ถูกแก้ไขในVersionนั้นของGitถึงจะถูกเพิ่มเข้าไปใน.Git directory

* Three stage

git มี3 ส่วน



* Working directory

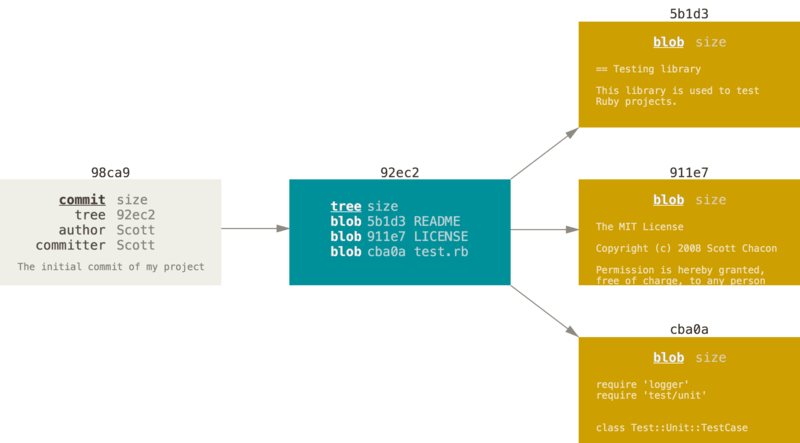
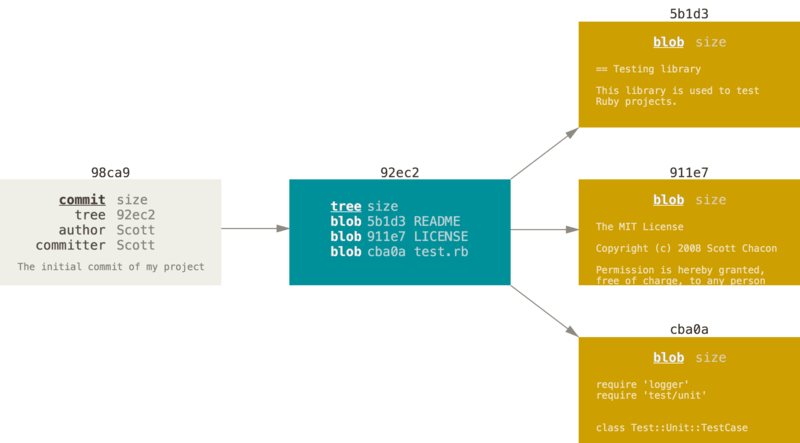
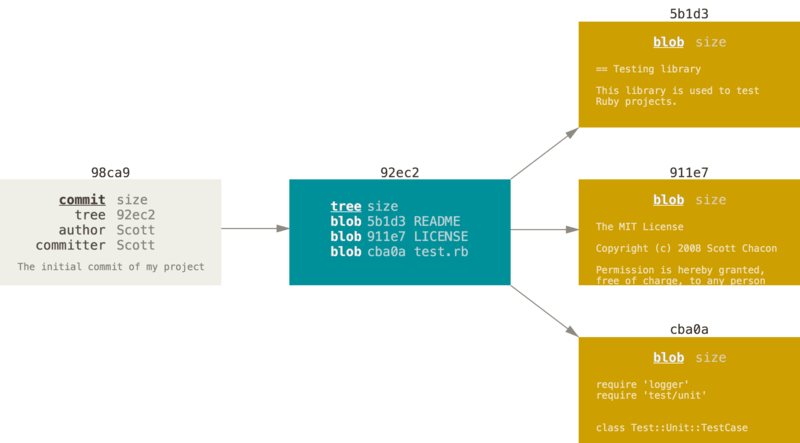
คือส่วนที่เราเปลี่ยนแปลงFileต่างๆที่เกี่ยวกับProject (ส่วนมากคือ Project directory) และข้อมูลในส่วนนี้เองที่Gitจะนำไปใช้ในStaging areaต่อไปและ

ถ้าเราไปดึงGit directoryมาจากที่อื่นGitก็จะสร้างFileในส่วนนี้จากข้อมูลที่อยู่ในGit directory

* Staging area

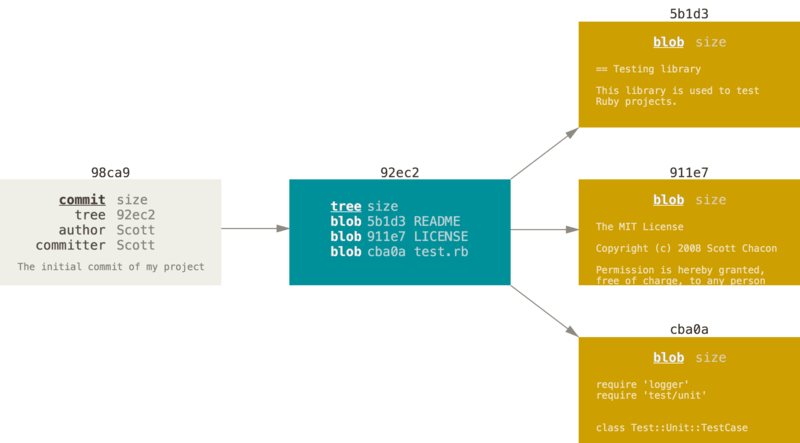
คือที่เก็บBlobที่ถูกสร้างขึ้นจากส่วนWorking directoryและเป็นส่วนที่จะถูกดึงข้อมูลไปใช้ใน.git directoryต่อไป

* Blob คือfileในWorking directory ที่ถูกChecksumและค่าที่ได้จากการChecksumจะนำไปแทนชื่อไฟล์

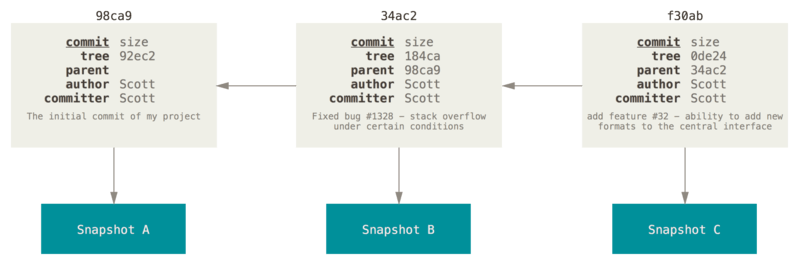


* .Git directory (repository)

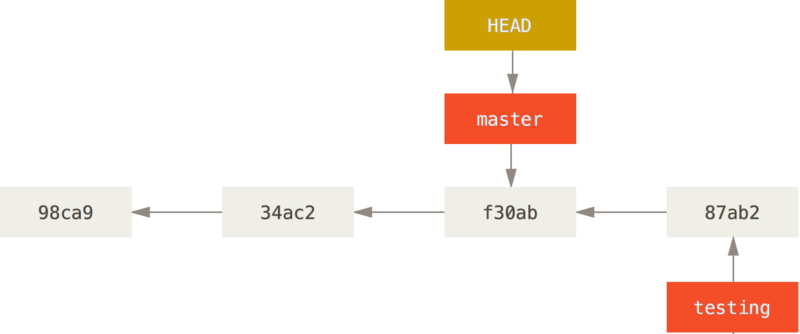
คือส่วนที่เก็บCommit objectที่GitดึงมาจากStaging area Gitจะสร้างCommit object ประกอบไปด้วยFile checksum ของแต่ละProject subdirectory(Project tree) และ File commitที่จะชี้ไปยัง Project root folder



ถ้าเราCommitต่อไปFile pointerอันใหม่ก็จะชี้ไปที่อันเก่าไปเรื่อยๆ

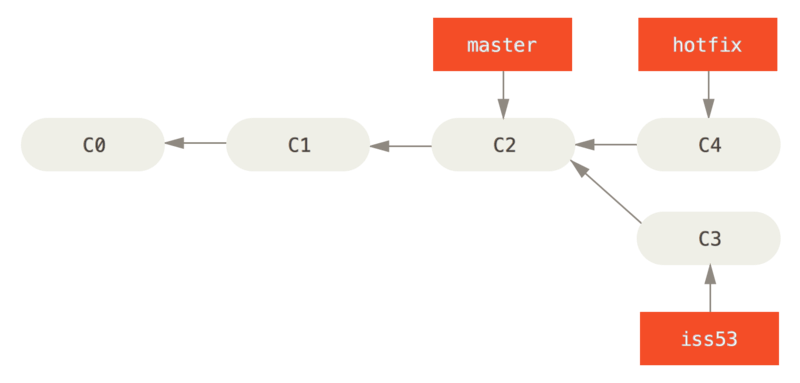


การBranchingจะใช้Pointer ชี้ไปที่Commitนั้น MasterคือBranch default name ที่ถูกสร้างขึ้นอัตโนมัติและเป็นBranchที่จะใช้เพื่อสร้างProject Headคือตัวชี้Branchที่เราอยู่ขณะนี้ Testingคือ ชื่อ(สมมุติ)Branchที่สร้างขึ้นใหม่

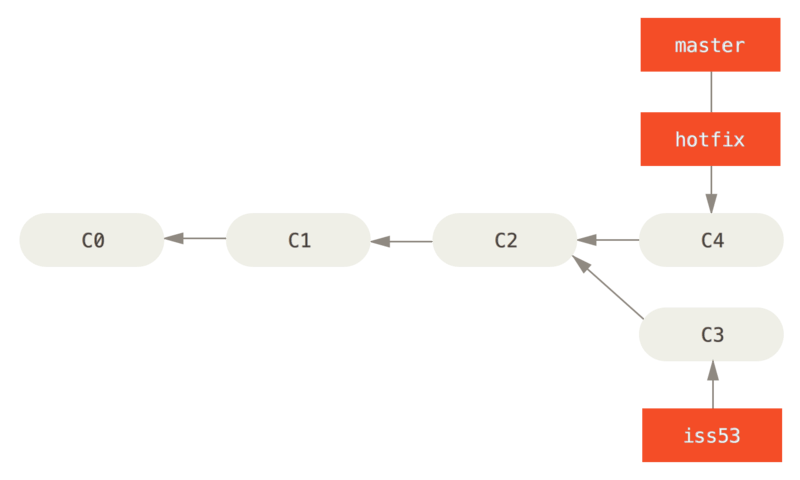


* Merge

ในกรณีที่เราสร้าง Branchที่ชื่อ hotfix ซึ่งเป็นbranchที่ต่อจากmaster

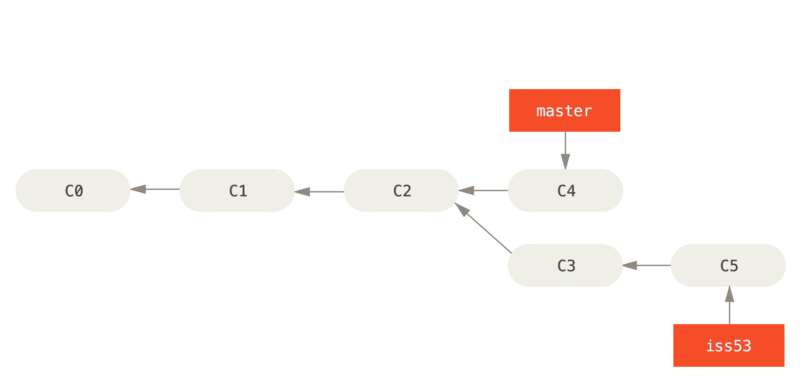


เมื่อ merge hotfix เข้าMaster pointer master จะเลื่อนไปข้างหน้าชี้ที่เดียวกับ hotfix

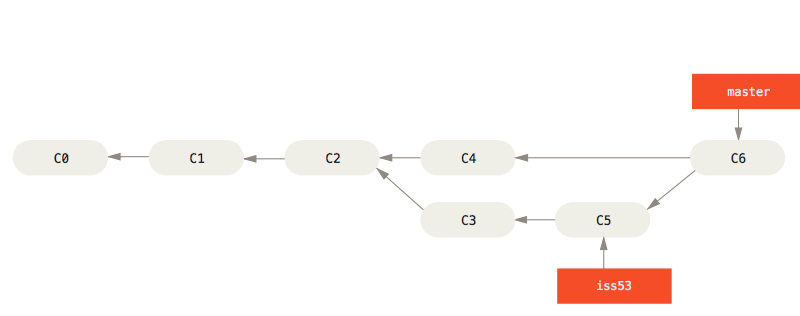


เมื่อmergeเสร็จเราจะต้องลบbranch hotfix ออกเพื่อลบpointer hotfix ซึ่งเป็น pointerที่ทำงานซ้ำซ้อนกับmasterออก

ในกรณีที่ต้องการจะmerge branch iss53 ซึ่งเป็นbranchที่ไม่ได้ต่อจากmaster



Git จะทำการสร้างcommitอัตโนมัติและชี้ไป2ที่ ที่masterเก่าอยู่และbranchที่จะmergeเข้า



* push

* pull

Thank to: <https://git-scm.com/docs/>

ณภัทร จารุพันธ์